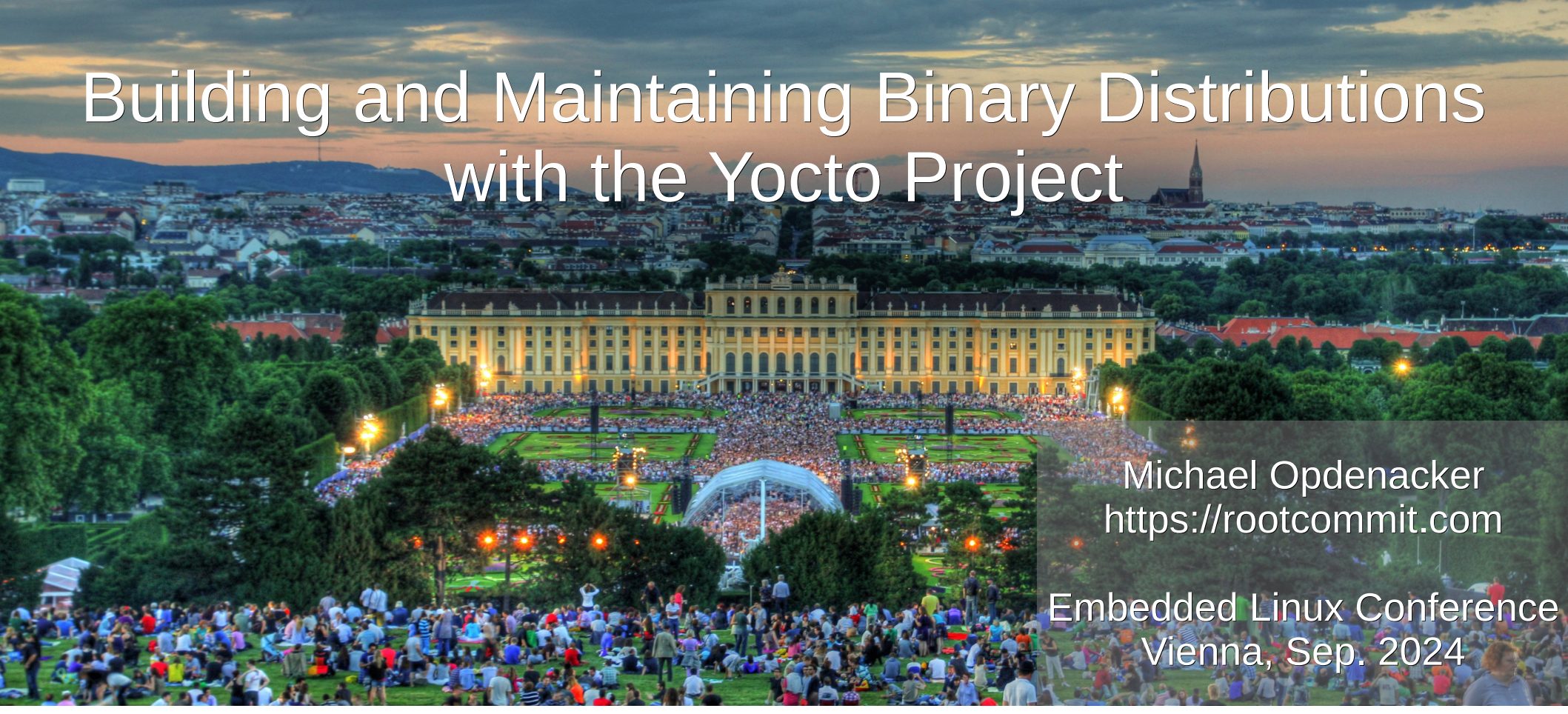
Building and Maintaining Binary Distributions with the Yocto Project



Michael Opdenacker
https://rootcommit.com

Embedded Linux Conference
Vienna, Sep. 2024

EMBEDDED LINUX CONFERENCE

yocto PROJECT

root commit

Michael Opdenacker

- Embedded Linux engineer, trainer and consultant
- STMicroelectronics, Texas Instruments, Bootlin (founder) and now Root Commit (founder).
- Contributor to OpenEmbedded / Yocto, Linux kernel, Elixir Cross Referencer...
- Passionate about learning and sharing knowledge
- Never missed any ELC Europe
- Except yesterday morning 🤨 🌊

root commit

About Root Commit

- First announced in this room, a few seconds ago 😁
- Mostly ready: preliminary website, LWN.net subscription...
- Goals:
  - **Contribute** to Free and Open Source projects, do technical research and **share experience** through technical articles and presentations at ELC.
  - Offer **embedded Linux training sessions** with innovative teaching techniques, to help you commit to **remembering what you learn.**
  - Offer embedded Linux **consulting services.**
- https://rootcommit.com

root commit

My involvement in OE / Yocto

- Got started with OpenEmbedded in 2024 with the earliest contributors: Mickey Lauer, Philip Blundell, Koen Kooi...
- Between 2006, got busy with some other projects.
  Buildroot and manual building were good enough.
- Served for 3 years as documentation maintainer for the Yocto Project.
- The Yocto Project got funding from the Sovereign Tech Fund for improving binary distribution support.
- Disclaimer: I do not represent the Yocto Project, which decisions are made by its members through its Technical Steering Committee. I’m just a contributor!

root commit

Goals of this presentation

- Share what I learned about binary distros
- Cover recently added features and possible future developments.

root commit

What I learned

root commit

Binary distros vs custom root filesystems

Binary distros

- Meant to be updated through packages.
- Updates are as small as they can be and one root partition is enough.
- Enable to remove packages
- Enable to add packages thanks to *package feeds*
- Need to support configuration at run time
- Can be updated without rebooting in most cases.

Custom root filesystems

- Meant to be updated through an entirely new image.
- This consumes more bandwidth and is less space efficient (need A/B partitions)
- Don’t enable to remove or add applications
- Typically customized by the build system at generation time.
- Cannot be updated without rebooting

root commit

Where each shines

Binary distros

- **General purpose** desktop, server and cloud distros (Debian, Fedora, Ubuntu…), meant to be configured at run time.
- Some embedded devices where the root system is updated separately from applications (e.g. some carmakers).

Custom root filesystems

- **Dedicated** embedded systems, in which you want no unused components, and which are not meant to be configured at run time.
- Systems which have to go through full validation at each update.

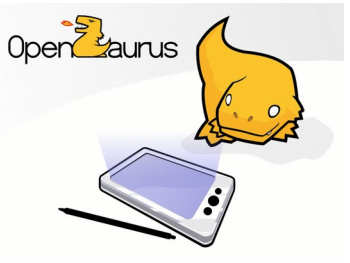
root commit

OE and Yocto binary distros (1)

- OpenEmbedded was created to generate the OpenZaurus distro for Sharp Zaurus devices.
- Ångström Distribution generated by OpenEmbedded supporting many devices. Maintained by Koen Kooi.

  Ångström was great to quickly boot a new device and deploy demos and add extra packages without needing to run the build system.

  Just boot a custom built Linux kernel and boot it with an Ångström root filesystem.
- Leider ist Ångström met dem 71er gefahren (wie man sagt in Wien)
- Ångström is defunct since 2015 🪦



root commit

OE and Yocto binary distros (2)

- Poky is used as a reference distribution for the Yocto Project to test OpenEmbedded builds.

  However, generated images didn’t have a package management system.

- Yoe Distro
  An attempt to have a new Yocto generated binary distribution, maintained / started by Khem Raj.
  https://www.yoedistro.org/

  However, doesn’t ship binary images and packages yet.

YOE

root commit

Standard distros vs Yocto built ones

Standard distros

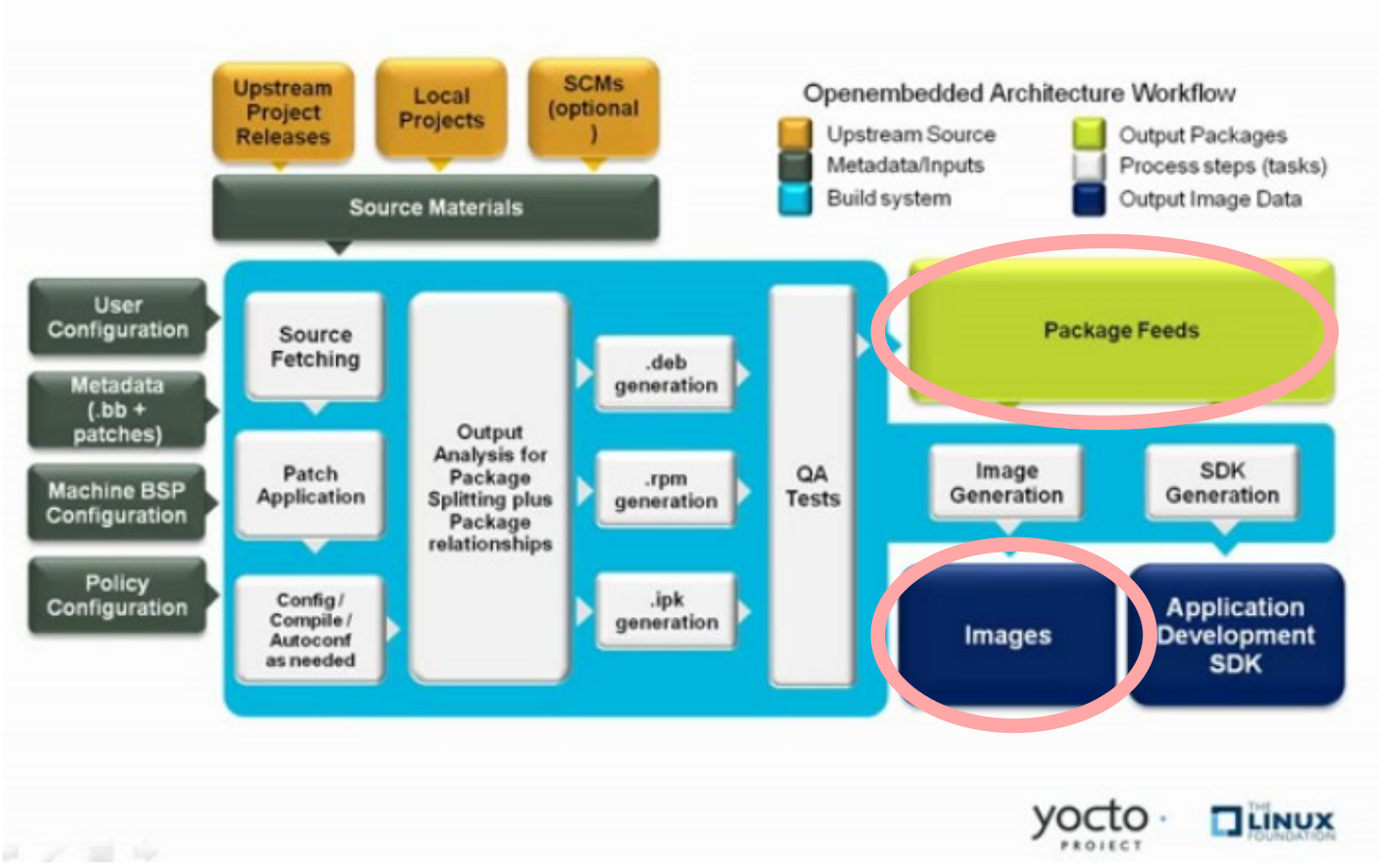
- Pre-compiled
- Limited ability to customize
- But faster security updates
- Commercial support available

Yocto built ones

- Need to build them from source
- Completely customizable
- But slower security updates (thanks to community contributions and member funding).
- Commercial support available too (e.g. Wind River)
- The Yocto project doesn’t ship binary feeds yet.

root commit

OpenEmbedded Binary Package Generation



root commit

Advantages using binary packages

- OK, binary packages can be used to add applications to a root filesystem
- But they can also be used to remove an application or a set of files.
- Buildroot, not using packages, cannot do that and has to rebuild from scratch when something has to be removed.

root commit

Available package formats

Rpm

- Users: Fedora, Red Hat, Poky (by default)
- Low level tool: `rpm`
- Front-end: `yum`
- Test:
  1.3 GB of packages
  45 MB .tar.bz2 image

Deb

- Users: Ubuntu, Debian, Poky (non-default)
- Low-level tool: `dpkg`
- Front-end: `apt`
- Test:
  1.1 GB of packages
  27 MB .tar.bz2 image

Ipk

- Users: OpenWRT, Poky (non-default)
- Simplified version of Deb
  Maintained by the Yocto Project
- Only one tool: `opkg`
- Test:
  1.8 GB of packages
  19 MB .tar.bz2 image

Test: core-image-minimal, poky master (Sep. 11, 2024), with package management

root commit

Why are `ipk` packages bigger than `deb` ones?

```
> find . -name "btrfs-tools[-_]6*" -exec du -sh {} ';'

808K   ./deb/core2-64/btrfs-tools_6.10.1-r0_amd64.deb
884K   ./rpm/core2_64/btrfs-tools-6.10.1-r0.core2_64.rpm
1,1M   ./ipk/core2-64/btrfs-tools_6.10.1-r0_core2-64.ipk
```

Let's extract the contents of the `.deb` and `.ipk` files with `ar -x`, to `deb/` and `ipk/` directories:

```
> tree --du --charset=ascii deb ipk
[     829304]  deb
|-- [        780]  control.tar.xz
|-- [     824424]  data.tar.xz
`-- [          4]  debian-binary
[    1118594]  ipk
|-- [        710]  control.tar.gz
|-- [    1113784]  data.tar.zst
`-- [          4]  debian-binary
```

`ipk` package contents are compressed with `zstd`, not as powerful as `xz`, but much less CPU intensive.

That's better for embedded systems.

root commit

Choosing the package format

- Set `PACKAGE_CLASSES` in `conf/local.conf`

```
PACKAGE_CLASSES ?= "package_deb"
PACKAGE_CLASSES ?= "package_deb package_ipk package_rpm"
```

- Though packages are generated for all `PACKAGE_CLASSES`, only the first setting is actually used to generate the image.

root commit

Enabling package management

- Though OpenEmbedded uses packages to install applications and other files, by default there is no package manager on Poky's `core-image-minimal` image.
- If you want to be able to use package management at run time:
  - Add to `conf/local.conf`:

    ```
    EXTRA_IMAGE_FEATURES += "package-management"
    ```

  - Or to an image recipe:

    ```
    IMAGE_FEATURES += "package-management"
    ```

- See EXTRA_IMAGE_FEATURES and IMAGE_FEATURES.

root commit

Create a package feed

- Package feeds are created automatically in `tmp/deploy/[rpm|deb|ipk]` when you build an image.
- ⚠️ Achtung: the package indexes (catalogs of packages and versions) are not created by default. You need to create them with:

```
> bitbake package-index
```

root commit

Publish a package feed

- Your package feed contents are in `tmp/deploy/<format>`
- You may copy that to a directory shared by a web server
- For development and testing, the quickest way is to run a local server from the command line. No need to set up an Apache server! 😉

```
> cd tmp/deploy/ipk/
> python3 -m http.server
```

This starts an HTTP server on local TCP port 8000

root commit

Use a package feed

- You need to configure the package manager in the image to let it know the HTTP(S) server details.
- Set the `PACKAGE_FEED_URIS`, `PACKAGE_FEED_BASE_PATHS`, and `PACKAGE_FEED_ARCHS` variables in `conf/local.conf`

root commit

Use a package feed - Example

```
PACKAGE_FEED_URIS = "https://example.com/packagerepos/release \
                     https://example.com/packagerepos/updates"
PACKAGE_FEED_BASE_PATHS = "rpm rpm-dev"
PACKAGE_FEED_ARCHS = "all core2-64"
```

Given these settings, the resulting package feeds are as follows:

```
https://example.com/packagerepos/release/rpm/all
https://example.com/packagerepos/release/rpm/core2-64
https://example.com/packagerepos/release/rpm-dev/all
https://example.com/packagerepos/release/rpm-dev/core2-64
https://example.com/packagerepos/updates/rpm/all
https://example.com/packagerepos/updates/rpm/core2-64
https://example.com/packagerepos/updates/rpm-dev/all
https://example.com/packagerepos/updates/rpm-dev/core2-64
```

root commit

Image and package feed contents

- What goes into the image?
  - The list of packages defined by `IMAGE_INSTALL` and the image that you build:

```
> bitbake core-image-minimal
```

- What goes into the package feed?
  - The list of packages that you build:

```
> bitbake hello
> bitbake world
> ...
```

root commit

Package managers: quick reference

Rpm

- Configuration: `/etc/yum.repos.d/`
- Commands:
  ```
  dnf update
  dnf install
  dnf remove
  dnf upgrade
  ```

Deb

- Configuration: `/etc/opkg`
- Commands:
  ```
  apt update
  apt list --upgradable
  apt upgrade
  ```

Ipk

- Configuration: `/etc/opkg`
- Commands:
  ```
  opkg update
  opkg install <package>
  opkg remove <package>
  opkg upgrade –noaction
  opkg upgrade
  ```

root commit

PR value

- PR = *Package Revision*
- Only needed when applying package updates
- Example:
  - Currently installed package: `myapp-1.0-r0`
  - Available bugfix update: `myapp-1.0-r1`
- This makes sure that the update prevails and gets installed. Not necessary when there is a version number increase.

1.0-r0
PV-rPR

https://docs.yoctoproject.org/ref-manual/variables.html#term-PR

root commit

PR Server

- A PR server is a process which increases the PR (revision) value when a new package output hash is found.
- Therefore, also needs a Hash Equivalence Server to work properly.
- Can either be a local server:
  `PRSERV_HOST = "localhost:0"`
- Or a server shared by multiple builders:
  `PRSERV_HOST = "192.168.1.17:8585"`

https://docs.yoctoproject.org/dev-manual/packages.html#working-with-a-pr-service

root commit

Newly added features

root commit

Building from source

Maintain product ……..

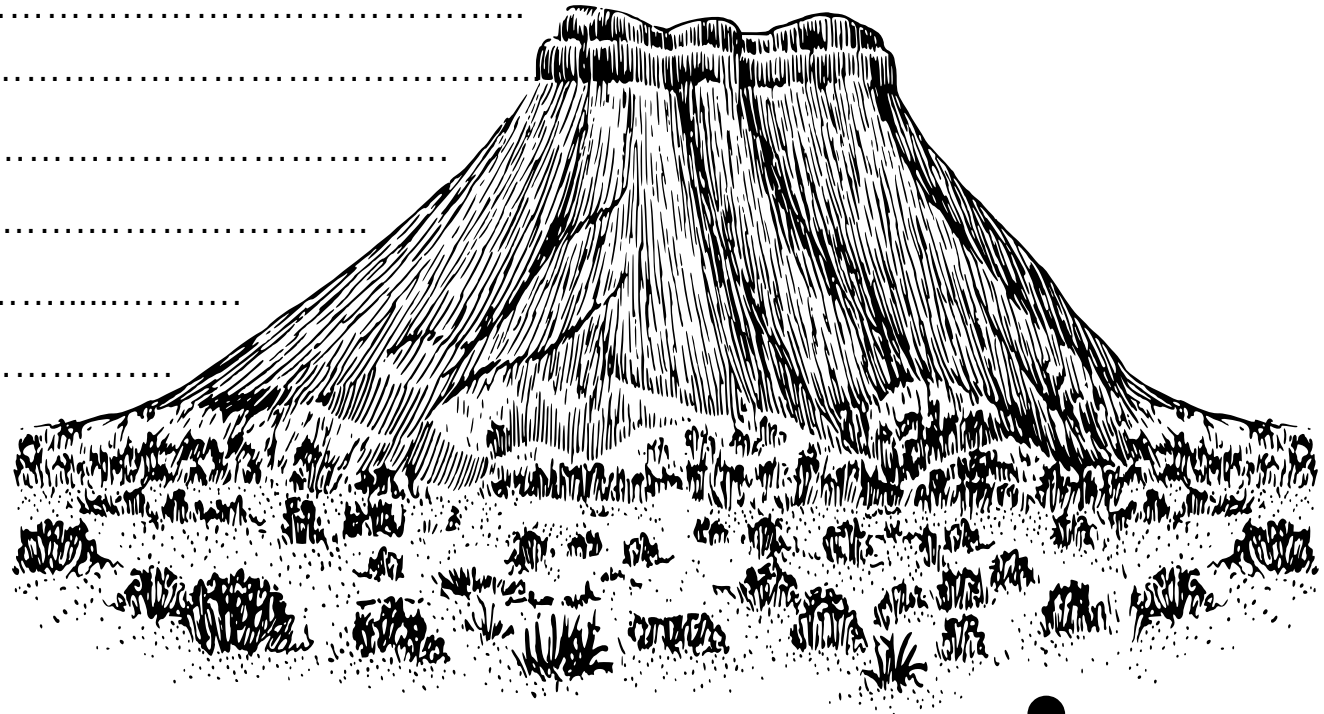
Debug system and application……

Customize distro and image……

Create, customize BSP layer……..

Create layer(s), add recipes……..……

Default settings, image, and distro……



Quite steep to climb!

root commit

Using a Yocto binary distribution

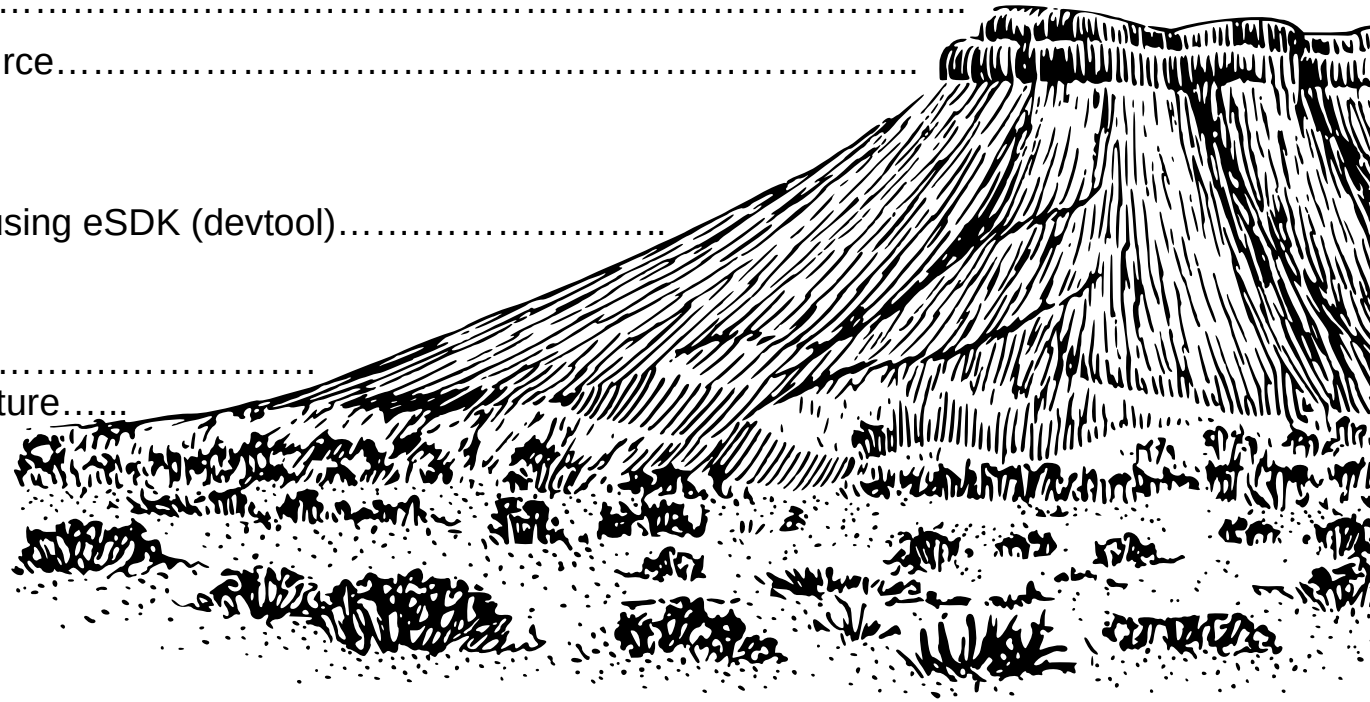
Maintain product ……..……..

Full system optimization, building from source……..

Tweak packages, build new applications, using eSDK (devtool)……..

Add ready-made packages to the image……

Use a ready-made image for your architecture…...



Easier to get started!

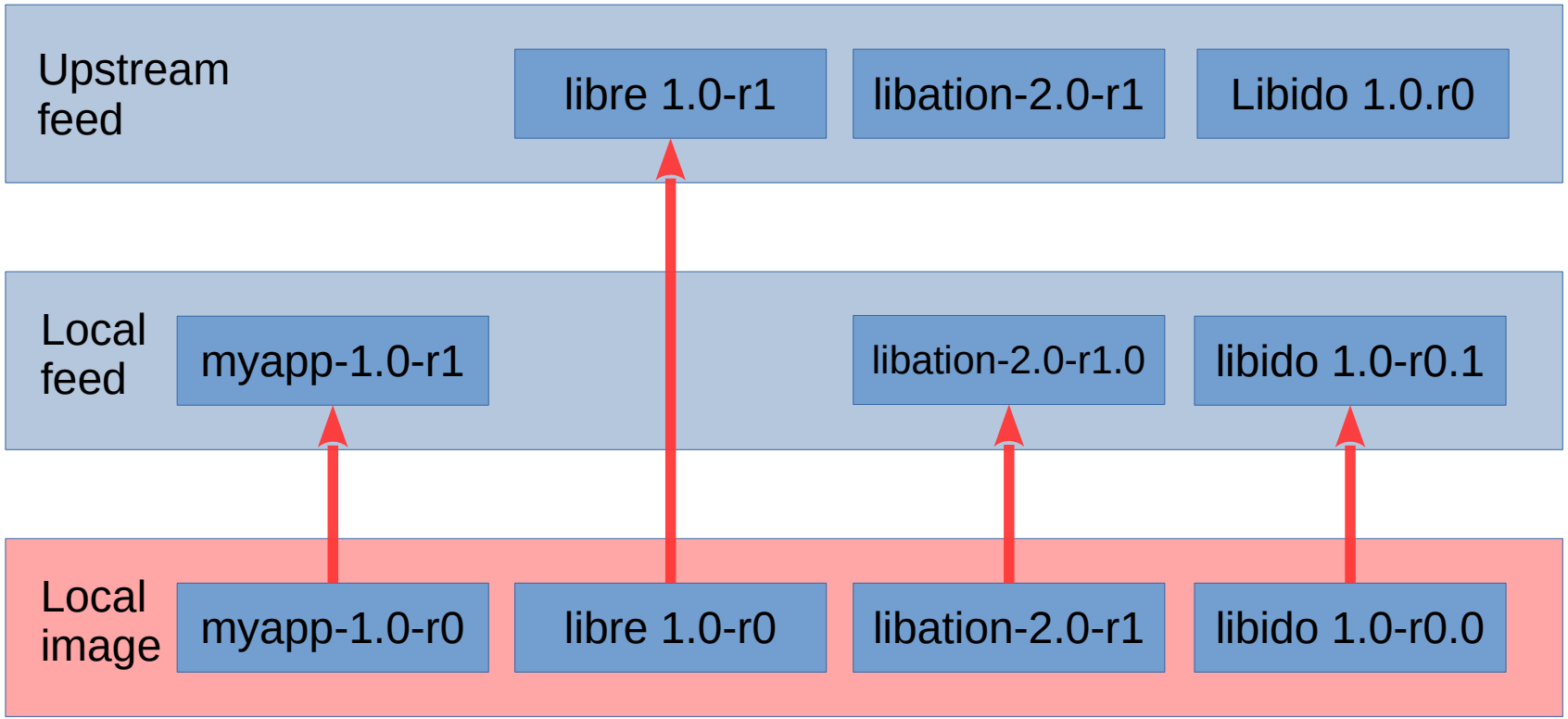
root commit

Improvements to the PR server

- Overhaul of the PR server Python code (`bitbake/lib/prserv`), aligning with Hash Equivalence server (`bitbake/lib/hashserv`)… Merged in *Scarthgap*.
- Implement new *read-only* mode
  Useful for an upstream distro to share PR information publicly, without having to implement authentication.
- Implement support for an *upstream* PR server.
- Features available in `Styhead (5.1)`

root commit

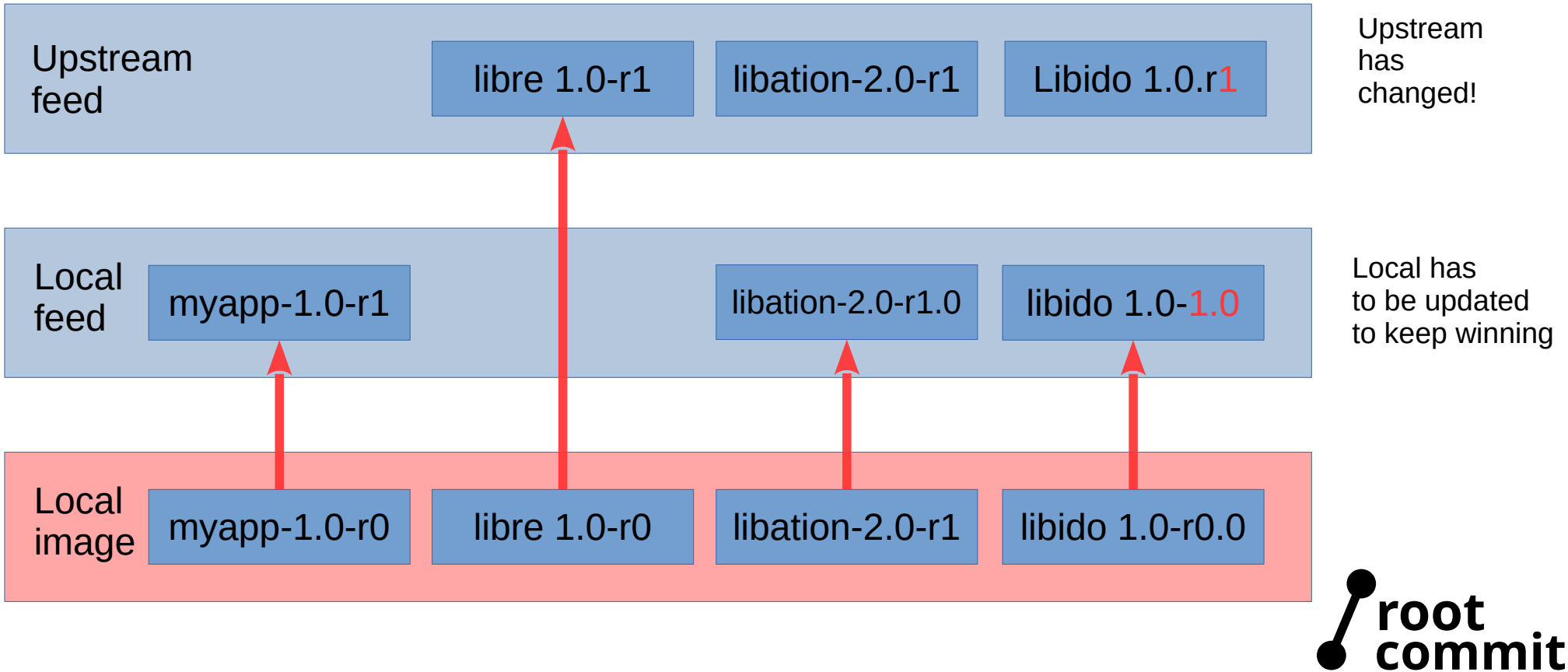
Usefulness of upstream PR server



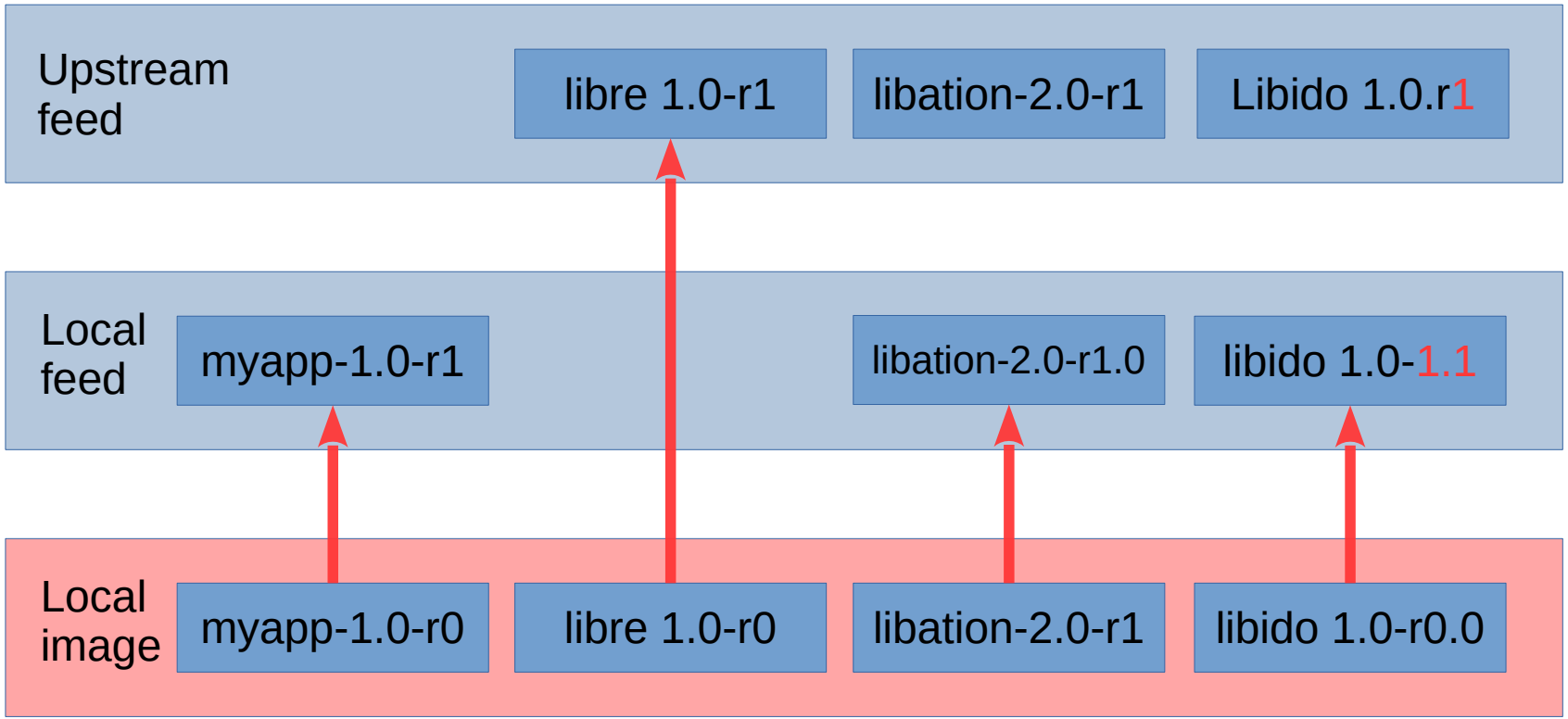
Revisions enable local customizations of upstream packages

root commit

Usefulness of upstream PR server



Usefulness of upstream PR server



One more local change

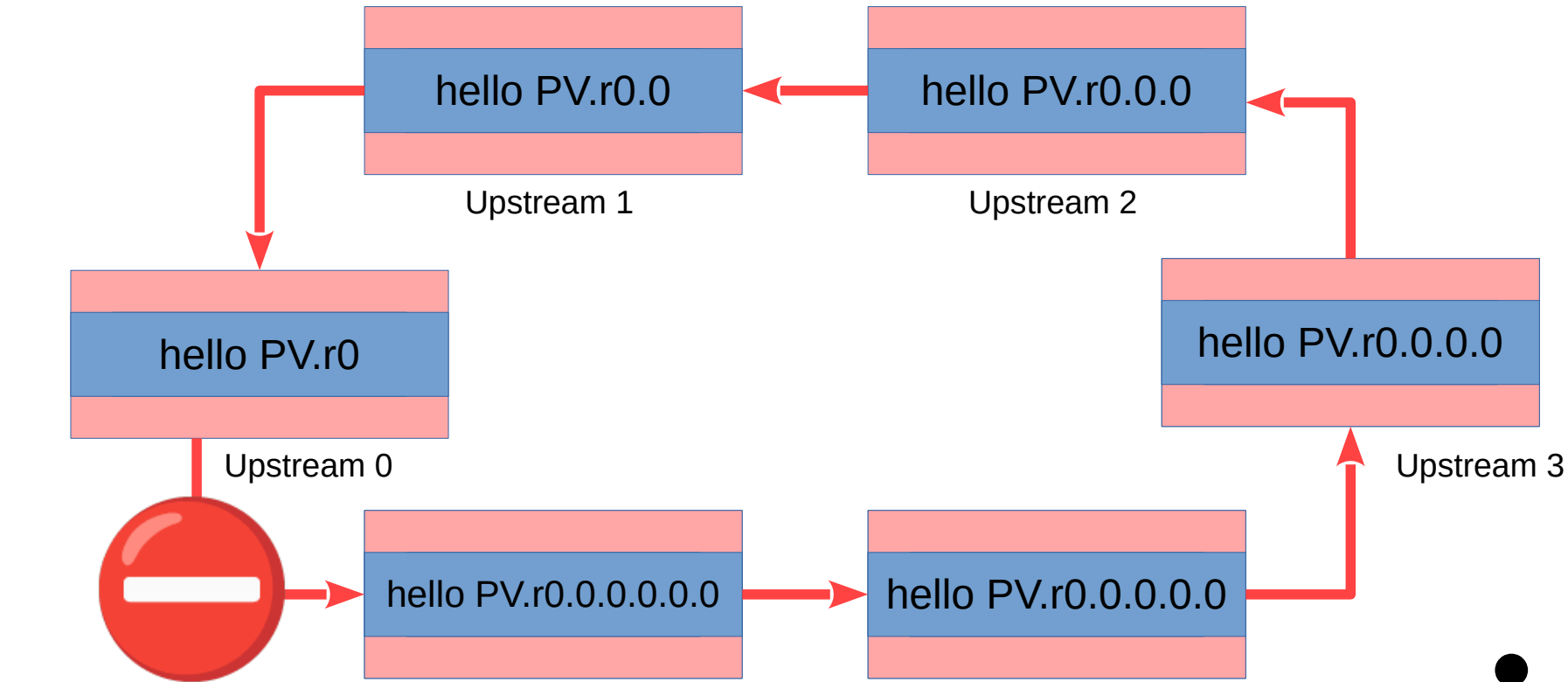
root commit

Understand revision increasing logic

- Comparing versions is more complex than just string comparison (e.g. `1.20` > `1.3`)
- Read the BitBake selftests in bitbake/lib/prserv/tests.py to understand the expected logic.
- Designed to make upstream always prevail.
- An additional feature available in the code, but no used yet:
  - *History mode*: allow to decrease a PR number if the local output hash has previously been used.
  - *No history mode*: never decrease a PR number (default setting)

root commit

Nested upstream PR servers!



Oops, forgot
to forbid infinite loops! 😱

root commit

How to use an upstream server

- Start the local server from the command line:

```
> bitbake-prserv [-h] [-f FILE] [-l LOG] [--
loglevel LOGLEVEL] [--start] [--stop] [--host HOST]
[--port PORT] [-r] [-u UPSTREAM]
```

- Use the `PRSERV_UPSTREAM` variable (`conf/local.conf`)

```
PRSERV_HOST = "localhost:0"
PRSERV_UPSTREAM = “192.168.1.17:8585”
```

root commit

Testing package feed updates

- Oe-selftest implemented to enable the autobuilder to test upgrading an image from a previous release to the current version using a package feed.
- Implementation submitted here: https://lore.kernel.org/openembedded-core/20240429152221.3405405-1-michael.opdenacker@bootlin.com/T/#u
- Still need to separate the Poky specific parts to get the test merged in OE-core.

root commit

Preparing for future development

- Defined the scope of a binary distro prototype (which recipes and machines to target): https://wiki.yoctoproject.org/wiki/Binary_Distro_Prototype#Scope_of_a_Yocto_Binary_Distribution_Prototype
- Richard Purdie and the TSC proposed policies / requirements on how a binary reference distro for the project would behave, and for including new recipes and for covering new platforms or architectures: https://wiki.yoctoproject.org/wiki/Binary_Distro_Prototype#Policies_and_Processes

root commit

A missing feature!

- Ability to update the system SPDX description after installing extra packages or package updates.
- The current way SPDX is generated doesn't allow to generate `-spdx` packages, which would have made this possible.
- See this discussion: https://lists.openembedded.org/g/openembedded-architecture/message/1855

root commit

What to remember

- OpenEmbeded / Yocto has always been able to generate binary distributions, like Ångström (🪦)
- Thanks to funding from Sovereign Tech Fund, Yocto Project has made progress to being ready to release its own autobuilt binary images and corresponding package feeds.
- This will make it easier to get started with Yocto without having to compile from source.
- However, for this to happen, more funding (typically from project members) will be necessary.

root commit

Key features

- Choose a package format: `PACKAGE_CLASSES ?= "package_deb"`
- Add package management to your image: `EXTRA_IMAGE_FEATURES += "package-management"`
- Generate package feed index: `bitbake package-index`
- Enable a PR server: `PRSERV_HOST = "localhost:0"`
- Add an upstream server: `PRSERV_UPSTREAM = “192.168.1.17:8585”`
- Start a custom PR server: `bitbake-prserv <opts>`

root commit

Useful resources

- Yocto Project Wiki:
  https://wiki.yoctoproject.org/wiki/Binary_Distro_Prototype
- Yocto Project Manual
  https://docs.yoctoproject.org/dev-manual/packages.html#working-with-a-pr-service
  (doesn’t include the latest features yet)

root commit

Image credits

- Front page
  https://commons.wikimedia.org/wiki/File:Schoenbrunn_philharmoniker_2012.jpg
- OE and Yocto binary distros (1)
  https://en.wikipedia.org/wiki/OpenZaurus
  https://en.wikipedia.org/wiki/%C3%85ngstr%C3%B6m_distribution
- OE and Yocto binary distros (2)
  https://daddytypes.com/2008/01/02/poky_the_adorable_linux_mobile_build_platform_mascot.php
  https://raw.githubusercontent.com/YoeDistro/yoe-distro/master/docs/yoe-logo.png
- OpenEmbedded Binary Package Generation
  https://docs.yoctoproject.org/overview-manual/yp-intro.html#the-openembedded-build-system-workflow
- Building from source
  https://openclipart.org/detail/274343/table-mountain

root commit

Thanks!

- Thanks for attending!
- Thanks to Richard Purdie and Bruce Ashfield for their support during the project.
- Any questions or comments?
- Please test and contribute improvements!
- Slides available: Creative BY-SA v4
  https://gitlab.com/rootcommit/yocto-binary-distro

root commit